Return forecasting through ARMA extensions, ARCH & GARH

Business Intelligence per i Servizi Finanziari 2023-2024

Antonio Candelieri

Testing for stationarity

- In practice, time series of asset prices (as well as of interest rates) are often **non-stationary** → because there is not a price value to which they revert.

- The **augmented Dickey-Fuller** test (**ADF** test) is a statistical test for non-stationarity of a time series.

- Formally, the ADF performs the following regression:

$$X_t = \alpha + \beta t + \varphi X_{t-1} + \theta_1 \Delta X_{t-1} + \theta_2 \Delta X_{t-2} + \cdots + \theta_{p-1} \Delta X_{t-p+1} + \boldsymbol{\varepsilon_t}$$

- where α is a constant, β is a coefficient on a trend, and p refers to the number of lags used in the model
- The constraint $\alpha = \beta = 0$ implies a **random walk**, where $\alpha \neq 0$ and $\beta = 0$ implies a **random walk with drift**
- The order *p* is usually decided using the **Aikaike Information Criterion** (**AIC**) and **Bayesian Information Criterion** (**BIC**).
- The ADF test check for the value of φ → if $\varphi = 0$ then non-stationarity, if $\varphi < 0$ then stationarity

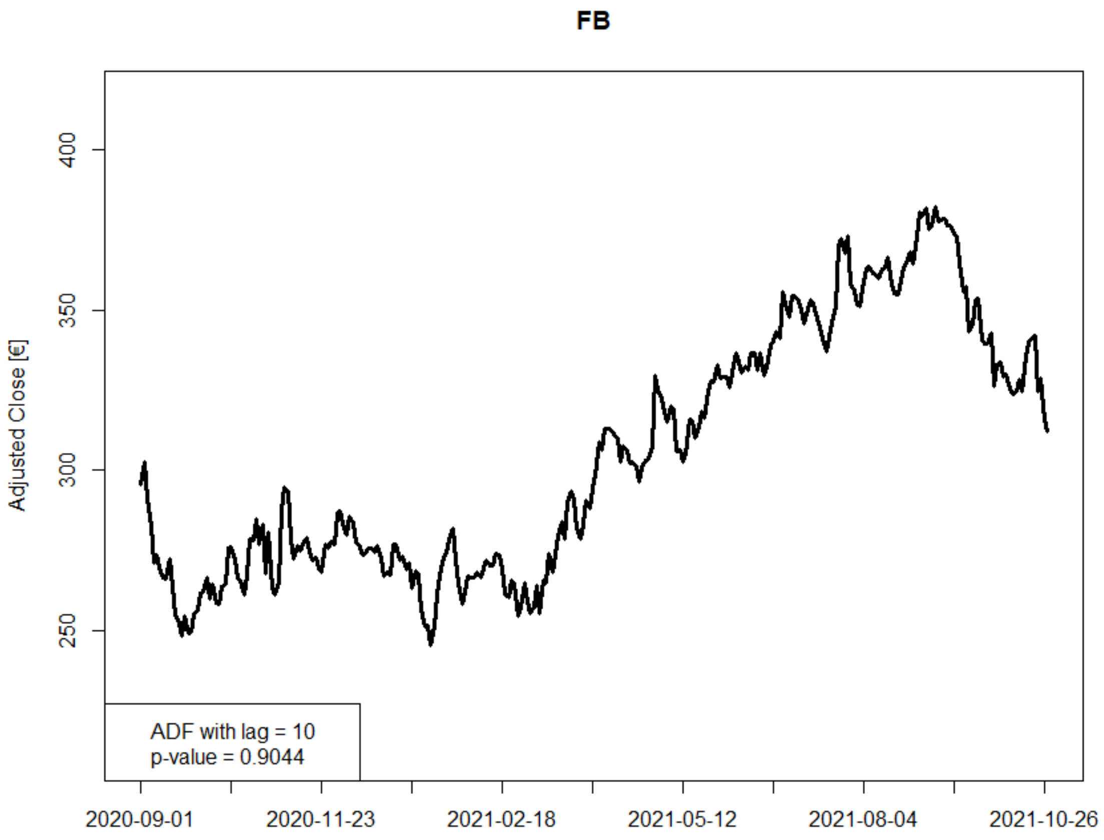
Handling non-stationarity

- Identifying the correct transformation to make the time series stationary is not always simple

- Some "heuristics" have been proposed:
  - Lag-1 autocorrelation close to zero or negative → no need for higher-order differencing
  - Positive autocorrelation up to 10+ lags → the series probably needs higher-order differencing
  - Lag-1 autocorrelation < -0.5 → the series may be over-differenced
  - Slightly over- or –underdifferencing can be corrected ith AR or MA terms

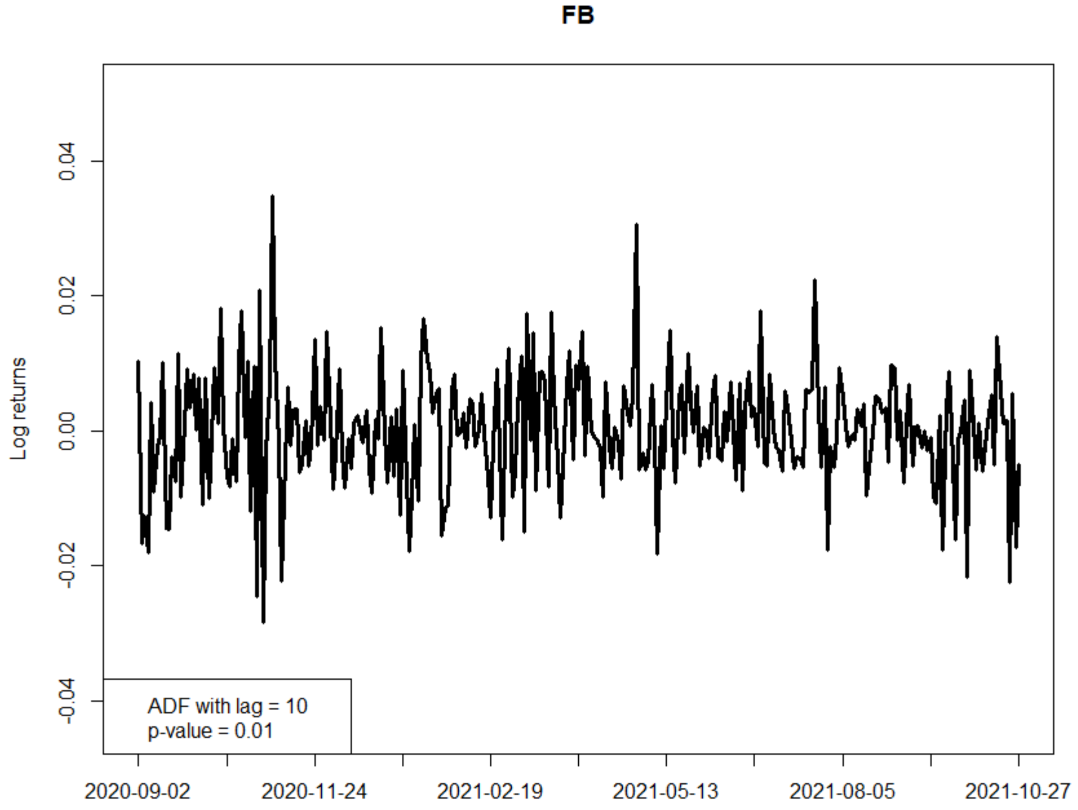
Testing for stationarity

As an example, the asset price time series reported in the picture is **non-stationary** (i.e., ADF *p*-value>0.05)



Testing for stationarity

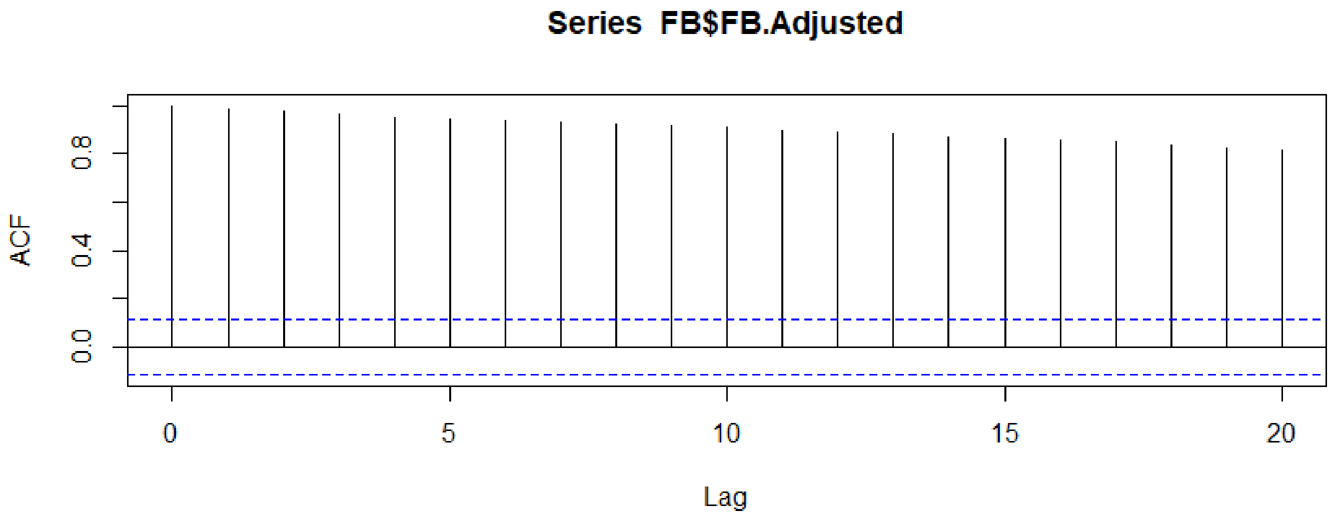
- Remind that the asset log-return series is given by: $r_t = p_t - p_{t-1} = \log P_t - \log P_{t-1} = \log \frac{P_t}{P_{t-1}}$
- If we consider this transformed series this is (usually) stationary (ADF *p*-value<0.05)

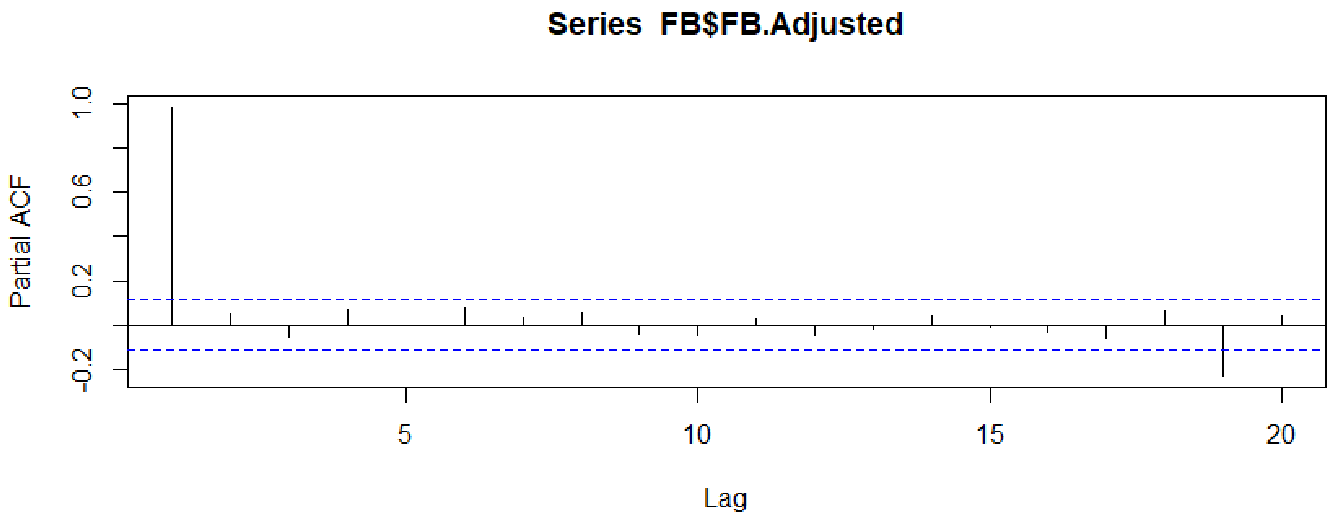


Testing for stationarity

- ARIMA is an extension of ARMA approach, to model autocorrelations in the data
  - AR(p) → autoregressive term with order *p*
  - I(d) → integrated term with order *d*
  - MA(q) → moving average with order *q*

- Plotting both the autocorrelation function (ACF) and the partial autocorrelation function (PACF) to identify significant autocorrelation at different lags.

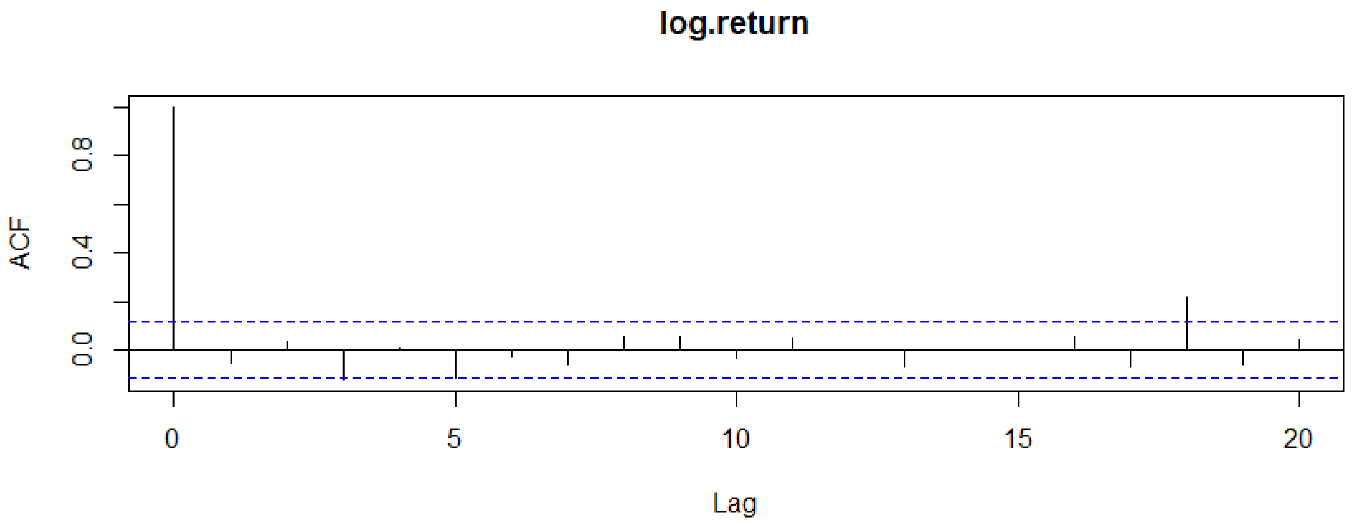


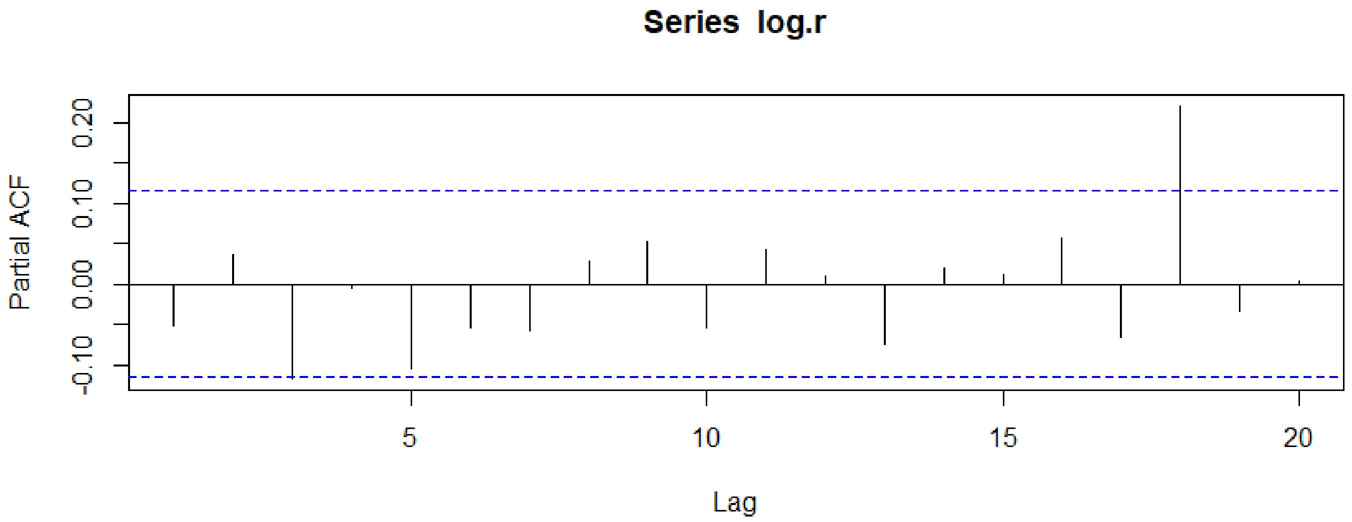


Testing for stationarity

- ARIMA is an extension of ARMA approach, to model autocorrelations in the data
  - AR(p) → autoregressive term with order *p*
  - I(d) → integrated term with order *d*
  - MA(q) → moving average with order *q*

- Plotting both the autocorrelation function (ACF) and the partial autocorrelation function (PACF) to identify significant autocorrelation at different lags.





How to identify the order(s)

For the AR term, if an AR model of order $\bar{p}$ is the correct model:

- the ACF will show a significant serial correlation up to lag $\bar{p}$.
- the PACF will not reflect correlation for lags beyond $\bar{p}$

For the MA term, if a MA model of order $\bar{q}$ is the correct model:

- the ACF will show significant coefficients for values up to lag $\bar{q}$.
- the PACF would show a similar behaviour (contrary to the AR model)

Since AR(p) and MA(q) terms interact, the information provided by ACF and PACF is no longer reliable and can only be used as a starting point. → Using validation (for instance measuring AIC or BIC)

ARIMAX & SARIMAX

ARIMAX: **ARIMA** with e**X**ogenous inputs

- It allows to add further input variables (e.g., sentiment indicators, information from fundamental analysis, …)
- Although it is anyway a linear regression model, it is more difficult to interpret

SARIMAX: **S**easonal **ARIMAX**

- For time series with seasonal effects, we can include AR and MA terms modelling seasonality/periodicity
- The ARIMAX(p,d,q) becomes SARIMAX(p,d,q)x(P,D,Q)
- For instance, when using monthly data and seasonal effect length is 1 year, the seasonal AR and MA terms would reflect this specific relation (i.e., P=Q=12)

Forecasting volatility

- Predicting volatility is a crucial topic in finance

- Volatility is uasually not constant over time

- Changes in volatility (i.e., variance) implies challenges for time series forecasting through ARIMA, as they assume stationarity

- Two approaches to model volatility to predict changes in variance
  - the **ARCH** (autoregressive conditional heteroskedasticity) model
  - the **GARCH** (generalized autoregressive conditional heteroskedasticity) model

The ARCH model

- **Heteroskedasticity** is the technical term for changes in a variable's variance

- The **ARCH(*p*)** model is simply an AR(*p*) model which is applied to the variance of the residuals (i.e., errors) of a time series model

- This makes the variance at time *t* conditional on lagged observations of the variance

- Thus, the error terms ε_t are the residuals of a linear model such as ARIMA, and split into a time-dependent standard deviation σ_t and a disturbance z_t

The GARCH model

- The ARCH model is relatively simple, but often <u>requires many parameters</u> to capture the volatility patterns of an asset-return series

- A **GARCH(*p*,*q*)** model applies to a log-return series and assume an ARMA(p,q) model for the error of the variance terms ε_t

How to build a volatility forecasting model

- Developing a volatility forecasting model for an asset-returns time series consists of 4 steps:
  1. Build an ARMA model of the financial time series (by using ACF and PACF, or validation)
  2. Test the residuals of the model for ARCH/GARCH effects (again by using ACF and PACF, or validation, <u>for the series of the squared residuals</u>)
  3. Specify a volatility model if serial effects are significant, and jointly estimate the mean and volatility equations
  4. Validate the model carefully and refine it, if needed.

- When applying volatility forecasting to return series, the serial dependence may be limited so that a constant mean may be used instead of an ARMA model.

The ML4T workflow

- **ML4T** stands for **Machine Learning for Trading**: and end-to-end perspective of the process of ***designing***, ***simulating***, and ***evaluating*** a ***trading strategy driven by ML algorithms***

- A realistic simulation of a strategy needs to faithfully represents how security markets operate and how trades are executed
  - The institutional details of exchanges must be considered (e.g., available order types, price determination, …)
  - Accurate performance measurements
  - Methodological aspects to carefully consider in order to avoid biased results and false discoveries, leading to poor investment decisions

How to backtest a ML-driven strategy

- Source and prepare: ***market***, ***fundamental*** and ***alternative*** data
- Engineer predictive factors and features
- Design, tune, and evaluate ML models to generate trading signals
- Decide on trades based on these signals
- Size individual positions in the portfolio
- Simulate the resulting trades triggered using historical market data
- Evalute how the resulting positions would have performed

Some backtesting pitfalls and how to avoid them

- <u>**Getting the right data**</u>
  - **Look-ahead bias: use only point-in-time data.** We need to guarantee that conclusions are based only on point-in-time that does not inadvertently include information from the future
  - **Survivorship bias: track your historical universe.** This bias occurs when the backtest contains only securities that are currently active while omitting assets that have disappeared over time
  - **Outlier control: do not exclude realistic extremes**
  - **Sample period: try to represent relevant future scenarios**

- <u>**Getting the simulation right**</u>
  - Transactions costs
  - Timing of decisions

- <u>**Getting the statistics right**</u>
  - …being not too optimistic!